ModSecurity를 활용한 아파치 웹서버 보안 강화 가이드

KISA는 본 문서에서 언급한 ModSecurity 및 해당 도구 개발사인 Breach社와 어떠한 관계도 없으며, 국내 웹 해킹 피해 예방을 위해 공개 웹 방화벽인 ModSecurity를 보안 참고용으로 소개합니다.

2008. 06

KrCERT/CC 인터넷침해사고대응지원센터

KISA 한국정보보호진흥원 Korea Information Security Agency

1. 개 요

ModSecurity는 Apache 웹 서버에서 동작하는 오픈 소스 웹 방화벽이며, 소스의 재사용 및 재생산된 프로그램의 공개 조건인 GNU GPL을 따르는 공개 버전과 ModSecurity의 개발사인 Breach Security社의 상업용 버전이 있는데, 본 가이드에서는 공개 버전을 이용한 설치 및 운영 방법을 알아본다.

본 문서는 2006년 3월에 발간된 『ModSecurity를 이용한 아파치 웹서버 보안('06.3.14)』에 기초하여 그동안 업데이트된 내용에 대해 보강하고 2.x 버전의 내용을 추가하였다.

ModSecurity는 O'Reilly社에서 출간한 "Apache Security"라는 책을 쓴 Ivan Ristic가 개발한 툴로써, 설치 및 차단 Rule 설정 인터페이스가 CLI기반이어서 다소 불편하다는 단점은 있지만 그만큼 유연한 정책설정이 가능하고 입력 값 검증기능이 또한 매우 우수하다. Apache 웹서버는 전세계적으로 널리 사용되는 공개 웹서버이며, 많은 수의 중소기업에서도 사용하고 있다. 그러므로 고가의 상용 웹 방화벽 도입에 어려움을 겪는 중소기업에서는 ModSecurity를 활용하여 웹 사이트에 대한 보안수준 강화에 큰 도움을 얻을 수 있다. 하지만, 다수의 대형 웹서버를 운영하는 기업이나 복잡한 웹 환경을 운용하는 경우, 관리자 인터페이스 및 기술지원 측면을 고려한다면 상용 웹 방화벽 도입이 바람직하다.

본 가이드에서는 공개용 ModSecurity를 이용한 아파치 웹서버의 보안 강화방안을 살펴보고, 특히 국내에서 홈페이지 변조사고에 이용되고 있는 PHP Injection 공격에 대응하기 위하여 ModSecurity에서 어떻게 설정해야 하는지에 대해 사례를 중심으로 살펴보고, 웹서버의 가장 일반적인 공격인 SQL Injection, XSS 등의 대표적인 웹 공격에 대한 방어 방법도 함께 기술하였다.

ModSecurity는 웹 공격에 대한 침입탐지 및 침입방지 기능을 추가해 주는 아파치 웹서버의 하나의 모듈로 동작하며, 웹 클라이언트와 아파치 웹 서버 사이에 ModSecurity가 존재하여 클라이언트로부터 악의적인 접속요청이 발견되면 공격차단, 로깅 등 사전에 정의된 행위를 수행한다. 다른 아파치 모듈과 마찬가지로 ModSecurity를 아파치의 한 부분으로 설치할 수 있으며, 정상적으로 설치되었을 경우 ModSecurity의 추가적인 처리로부터 발생되는 부하는 거의 없다고 할 수 있다.

ModSecurity의 주요 특징은 다음과 같다.

o 요청(request) 필터링
- 클라이언트로부터 웹 요청이 들어올 때, 웹서버 또는 다른 모듈들이 처리하기 전에 ModSecurity가 요청 내용을 분석하여 필터링한다.

o 우회 방지 기술
- 경로와 파라미터를 분석하기 전에 정규화시켜 우회 공격을 차단한다.
- 즉, "//", "\/", ".", "%00" 등 우회 공격용 스트링을 제거하고, 인코딩된 URL을 디코딩한다.

o HTTP 프로토콜 이해
- 엔진이 HTTP 프로토콜을 이해하기 때문에 전문적이고 정밀한 필터링을 수행할 수 있다.

o POST 페이로드(payload) 분석
- GET 방식 뿐만 아니라 POST 메소드를 사용해서 전송되는 컨텐츠도 분석 가능하다.

o 감사 로깅
- POST를 포함하여 모든 요청의 모든 상세한 부분들까지 추후 분석을 위해서 로깅될 수 있다.
- ModSecurity에서 차단기능을 비활성화 시킨 후, 강력한 로깅 기능만으로 침입탐지 시스템 역할을 수행할 수 있도록 한다.

o HTTPS 필터링
- 엔진은 웹서버에 임베디드되어 있기 때문에 복호화 한 후에 요청 데이터에 접근하여 HTTPS를 통한 공격도 필터링할 수 있다.

공개 웹방화벽 사용자 커뮤니티

- **기술문서 오탈자 정보**
- **기술정보 및 최적화 등 정보공유**
- **기술문서 및 차단정책(룰) 배포**
- **사용자들간의 질의 답변**

http://www.securenet.or.kr > 열린지식 > 공개 웹방화벽 커뮤니티

2. ModSecurity 1.x 설치 및 운영

ModSecurity는 1.x 버전과 2.x 버전이 있다. 먼저 알아볼 것은 ModSecurity 1.x 버전이며 본 가이드에서는 아래의 환경에서 ModSecurity를 설치하여 테스트하였다.

o 커널 : Linux 2.4.20-8-686-smp
o 웹서버 : Apache 2.0.59
o ModSecurity 소스코드 디렉토리 : /usr/local/modsecurity-apache_1.9.5
o 아파치 소스설치 디렉토리 : /usr/local/apache2
o 아파치 웹서버 홈 디렉토리 : /usr/local/apache2/htdocs

Breach Security에서는 ModSecurity 1.9.5 버전을 마지막으로 1.x 버전에 대한 지원은 더 이상 없다는 뜻을 홈페이지를 통해 공지하였다.

2.1. ModSecurity 1.x 다운로드 및 설치

설치방법은 Linux에서의 과정을 위주로 살펴보겠다. 크게 2가지 설치방법이 있는데 소스를 통해 설치하는 방법과 바이너리 파일을 통해 설치하는 방법이 있다. 바이너리 파일을 통한 설치는 윈도

우즈 버전의 아파치를 사용하거나 컴파일러가 없을 경우에 사용하면 좋다. 본 가이드에서는 소스를 통한 설치 방법을 알아보도록 한다.

설치하고자 하는 ModSecurity 1.9.5 버전을 다음 사이트에서 다운로드 받을 수 있다.
http://www.modsecurity.org/download/modsecurity-apache_1.9.5.tar.gz

또는, KrCERT/CC 홈페이지 공개웹방화벽 페이지에서 다운로드 받을 수 있다.
http://www.krcert.or.kr/firewall2/index3.jsp 접속 후 modsecurity-apache_1.9.5.tar.gz 다운

다운로드 받은 후 다음의 명령으로 압축 및 패키징을 해제한다.

```
# tar xvzf modsecurity-apache-1.9.5.tar.gz
# cd modsecurity-apache-1.9.5; ls -al
linux-web:/usr/local# cd modsecurity-apache-1.9.2; ls -al

합계 88
drwxrwx---    6 1000     1000         4096  6월  7  2007 .
drwxr-xr-x   21 root     root         4096  1월 11 00:24 ..
-rw-rw----    1 1000     1000        28867  6월  7  2007 CHANGES
-rw-rw----    1 1000     1000          892  6월  7  2007 INSTALL
-rw-rw----    1 1000     1000        17989  2월  6  2007 LICENSE
-rw-rw----    1 1000     1000          993  5월 26  2007 README
drwxrwx---    2 1000     1000         4096  6월  7  2007 apache1
drwxrwx---    2 1000     1000         4096  6월  7  2007 apache2
drwxrwx---    3 1000     1000         4096  6월  7  2007 doc
-rw-rw----    1 1000     1000         1811  2월  6  2007 httpd.conf.example-minimal
drwxrwx---    2 1000     1000         4096  6월  7  2007 util
```

소스를 통한 설치 방법에도 웹서버 초기설치 시 웹서버 자체에 모듈을 설치하는 방식과 운영되고 있는 웹서버에 mod_security.c만을 컴파일하여 포함시키는 동적공유객체(DSO, Dynamic shared object) 방식 등 두 가지가 있다. 일반적으로 DSO 방식으로 설치하는 것을 권장하며 Static 방식으로 설치할 경우 아파치를 재설치 해야 하기 때문에 권장하는 방법은 아니다.

▷ DSO 방식 설치

DSO 방식은 아파치 웹서버의 재설치 과정 없이 기존에 운영되고 있는 아파치 웹서버에 모듈을 동적으로 추가하는 방식이므로 기존에 아파치 웹서버를 이미 운영 중이면서 DSO 설치방식을 지원하는 기관의 경우 이 방법을 택하길 권장한다. 그리고 설치하기에 앞서 유념해야 할 사항이 있는데, 필요한 경우 재차 거듭하겠지만, ModSecurity의 설치는 Apache의 버전에 의존적이다. Apache 1.x의 경우 ModSecurity 1.x버전만 설치가 가능하며 Apache 2.x의 경우 ModSecurity 1.x와 2.x 모두를 설치할 수 있다. 하지만 ModSecurity 2.x는 Apache2에만 설치가 가능하다.

① apxs를 이용하여 ModSecurity 모듈을 컴파일하고, 설치하고, 설정을 자동으로 변경한다.

```
# /usr/local/apache2/bin/apxs -cia /usr/local/modsecurity-apache_1.9.5/apache2/mod_security.c
```

위의 명령은 mod_security.c를 컴파일 하고(-c 옵션), 공유객체를 웹서버 modules 디렉토리에 설치하고(-i 옵션), 아파치 httpd.conf 설정파일에 적절한 LoadModule 줄을 추가(-a 옵션)한다. 참고로 apxs는 아파치 웹서버의 확장모듈을 컴파일하고 설치하는 도구로써, 여러 소스와 오브젝트파일을 LoadModule 지시어로 실행 중인 아파치 서버로 읽어 들일 수 있는 동적공유객체(DSO)를 만든다. 위의 결과로 modules 디렉토리에 mod_security.so가 생성되고 httpd.conf 파일에 "LoadModule security_module modules/mod_security.so" 행이 추가된다.

② 설치가 완료되면 다음과 같은 메시지가 출력이 될 것이다.

```
Libraries have been installed in:
   /usr/local/apache2/modules

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the `LD_RUN_PATH' environment variable
     during linking
   - use the `-Wl,--rpath -Wl,LIBDIR' linker flag
   - have your system administrator add LIBDIR to `/etc/ld.so.conf'

See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
chmod 755 /usr/local/apache2/modules/mod_security.so
[activating module `security' in /usr/local/apache2/conf/httpd.conf]
```

③ 아파치 웹서버를 재구동한다.
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start

여기까지 ModSecurity의 DSO방식을 이용한 모듈 설치가 끝났다. 실제 설치방법이 매우 간단하고 빠른 방법이다.

▷ 소스 컴파일을 통한 설치

DSO 방식이 아닌 정적으로 소스 컴파일 될 경우에는 ModSecurity 모듈이 웹서버의 body에 포함되게 된다. 이 방법은 DSO 방식에 비해 다소 실행 속도가 빠르지만, 아파치 웹서버를 재 컴파일 해야 하는 번거로움이 있다.

또한, 아파치 버전에 따라 설치를 위한 사전 설정을 달리 해 주어야 한다.

<아파치 1.x의 경우>

cd <apache1-source>
cp <modsecurity-source-path>/apache1/mod_security.c ./src/modules/extra

```
# ./configure --activate-module=src/modules/extra/mod_security - - enable-module=security
```

<아파치 2.x의 경우>

```
# cd <apache2-source>
# cp <modsecurity-source>/apache2/mod_security.c ./modules/proxy
# ./configure -enable-security --with-module=proxy:mod_security.c
```

아파치 1.x 또는 아파치 2.x에서 위의 과정을 거친 후에, 일반적인 아파치 컴파일과 설치 과정을 거치면 된다.

```
# make && make install
# /usr/local/apache2/bin/apachectl start
```

DSO 방식과는 달리 소스 컴파일을 통한 설치 시에는 httpd.conf 파일에 아무런 내용이 추가되지 않는다. DSO 방식과 마찬가지로 ModSecurity를 활성화시키기 위해서는, 다음 장의 ModSecurity 활성화 및 Rule 정의를 위한 환경설정이 필요하다.

2.2. ModSecurity 1.x 활성화 및 Rule 설정

ModSecurity를 설치하였다고 해서 바로 웹 방화벽 기능이 적용되는 것은 아니다. 이를 적용시키기 위해서는 아파치 웹서버 환경설정 파일(httpd.conf)의 <IfModule> 태그 안에 설정 지시자(directive)를 정의해 주어야 한다.

```
<IfModule mod_security.c>
   # mod_security configuration directives
   # 이하 생략...
</IfModule>
```

또는 ModSecurity를 위한 별도의 환경설정 파일을 만들고 이를 httpd.conf에 포함시킬 수 있다.

만약 "<apache_home>/conf/mod_security.conf" 라는 이름으로 ModSecurity 웹방화벽을 위한 환경설정 파일을 별도로 만들었을 경우 httpd.conf 파일에 다음과 같이 이 파일을 포함시켜 줄 수 있다.

Include conf/mod_security.conf

ModSecurity를 위한 Rule이 다양하고, 내·외부 웹 환경에 따라 Rule 변경 및 업데이트가 지속적으로 필요하기 때문에 별도의 파일을 이용하는 것이 좀 더 편리할 것이다.

ModSecurity는 다양한 기능의 설정을 위해서 상당히 많은 지시자들이 존재하는데, 이를 일일이 직접 작성하여 적용하기는 쉽지 않을 것이다. 따라서 아래 ModSecurity 홈페이지에서 제공하는 Rule template을 자신의 웹 환경에 맞게 수정하는 것이 용이할 것이다.

http://www.modsecurity.org/download/

위 페이지에서 "Download ModSecurity and Core Rules from Breach Security Network"를 클릭하여 들어가면 로그인 화면이 나온다. 사용자 계정이 없을 경우 간단한 등록절차 후 로그인 하면 Core Rule 뿐만 아니라 ModSecurity의 버전별 소스까지 다운받을 수 있다.

Core Rule에는 기본 설정 및 각 유형별 구체적인 Rule template이 있으므로 이들을 참고해 보자.

또는, KrCERT 홈페이지에서도 Sample Rule을 다운받을 수 있다.

http://www.krcert.or.kr/firewall2/index3.jsp 페이지 하단의 업체 및 버전별 Sample Rule 다운

여기에서는 <IfModule> 태그 안에 들어갈 지시자들 중 기본적인 지시자들을 알아보도록 한다. 상당히 다양한 지시자가 있으므로 자세한 것은 아래를 참고해 보기 바란다.

http://www.krcert.or.kr/firewall2/index2.jsp 에 modsecurity1.9.4-manual.pdf 다운

2.2.1 기본 환경설정

SecFilterEngine On

ModSecurity 기능을 활성화(enable) 시킨다.
- o On : ModSecurity 기능 활성화
- o Off : ModSecurity 기능 비활성화

SecFilterScanPOST On

POST 메소드의 payload를 점검한다.
ModSecurity는 다음과 같은 2가지 타입으로 인코딩된 Request body를 지원한다.

- o application/x-www-form-urlencoded (Form 데이터 전송 시 사용)
- o multipart/form-data (파일 전송시 사용)

다른 인코딩 타입은 대부분의 웹 어플리케이션에서 사용되지는 않는다.

**SetEnvIfNoCase Content-Type \
"^multipart/form-data;" "MODSEC_NOPOSTBUFFERING=Do not buffer file uploads"**

각 요청별로 POST payload 점검을 비 활성화할 수 있다. 환경변수 MODSEC_NOPOSTBUFFERING이 정의되어 있으면 POST payload 버퍼링을 하지 않는다.

SecFilterDefaultAction "deny,log,status:404"

룰이 요청에 일치하면 하나 또는 그 이상의 행위(action)가 발생된다. 각각의 필터별로 행위를 정의할 수 있지만 모든 필터들을 위한 기본 행위 집합을 정의하면 편리하다. 모든 필터에 적용될 수 있는 디폴트 행위는 SecFilterDefaultAction 지시자로 정의할 수 있다. 이 설정은 각 룰에 일치할 경우 접속요청을 차단하고, 로그를 남기긴 후 HTTP 상태코드 404를 보내는 예이다.

일반적으로 SecFilter로 정의한 옵션에 적용이 되며 다른 지시자와 옵션을 부가하여 짜여진 Rule은 SecFilterSignatureAction 으로 행위를 정의한다.

앞서 SecFilterDefaultAction 지시자로 필터링 규칙에 일치할 경우 기본적으로 어떤 행위(Action)를 하게 할 것인지에 대해 간단히 알아보았다. 그러면 필터링 규칙에 일치할 경우 일어날 수 있는 행위의 종류에 대해 알아보자. 먼저, 행위는 다음과 같은 3가지 종류로 나눌 수 있다.

구 분	설 명
Primary action	요청을 계속 진행할 것인지 차단 할 것인지를 결정하는 것으로, deny, pass, redirect 중 하나를 선택한다.
Secondary actions	Primary action에 의한 결정과는 독립적으로 수행되는 것으로 exec와 같은 몇 개의 secondary action들이 있다.
Flow actions	필터 룰의 흐름을 변경할 수 있는 것으로 다른 룰로 점프하게 하거나 몇 개의 룰을 건너 띄게 할 수 있다. Flow action에는 chain과 skipnext가 있다.

앞서 SecFilterDefaultAction 지시자에 의한 기본 적용 action의 예에서는 콤마(,)로 구분된 3개의 action을 정의하고 있다. 취할 수 있는 대표적인 행위는 다음 표와 같다.

행 위	설 명
pass	필터에 일치할 경우 요청을 그냥 허용한다. 이 action은 아무런 행위를 하지 않고 그냥 로그만 남겨 침입을 모니터링하거나 초기 환경설정 시 유용할 수 있다. 예) SecFilter KEYWORD "log,pass"
allow	pass에 비해 좀 더 강력한 행위로 이 action이 수행 된 후 다른 필터는 적용시키지 않고 곧바로 요청을 허용하게 된다. 예) SecFilterSelective REMOTE_ADDR "^192\.168\.2\.99$" allow 위의 예는 관리자 컴퓨터(192.168.2.99)에서의 접속은 항상 허용하도록 하고 있다.
deny	요청 처리를 차단한다. 상태 action(status)이 명시적으로 함께 사용되지 않으면 ModSecurity는 “HTTP 500 error code"를 반환한다.
status	요청이 거부되었을 경우 HTTP 상태를 제공한다. 예) SecFilter KEYWORD "deny,status:404"
redirect	필터가 일치하면 사용자를 주어진 URL로 이동 시킨다. 예) SecFilter KEYWORD "redirect:http://www.krcert.or.kr/warn.html"
proxy	필터가 일치하면 요청을 내부 리버스 프록시로 다시 쓴다. 예) SecFilter KEYWORD “proxy:http://www.example.com" ※ 이 동작은 mod_proxy가 반드시 설치되 있어야 한다.
exec	필터가 일치하면 특정 바이너리를 실행시킨다. 실행 될 파일은 전체 경로를 지정해 주어야 한다. 예) SecFilter KEYWORD "exec:/home/ivanr/report-attack.pl"
log	아파치 에러 로그(error_log)에 기록한다.
nolog	필터가 일치해도 기록하지 않으며, “audit logging"도 일어나지 않도록 한다.
skipnext	필터가 일치하면 하나 이상의 룰을 건너뛸 수 있다. 예) SecFilterSelective ARG_p value1 skipnext:2 SecFilterSelective ARG_p value2 SecFilterSelective ARG_p value3 (이후부터 룰이 적용된다.)

행 위	설 명
chain	두개의 룰을 연관 지어 허용하고 싶을 때 사용할 수 있다. 예) SecFilterSelective ARG_username admin chain SecFilterSelective REMOTE_ADDR "!^YOUR_IP_ADDRESS_HERE$"
pause	요청에 대한 응답을 하기 전에 정의된 수 milliseconds 동안 중지시킨다. 이는 웹 스캐너를 느리게 하거나 완전히 교란시킴으로써 스캔 공격을 억제시킬 수도 있다. 어떤 스캐너는 중지 시간이 너무 길면 스캐닝을 포기한다.
auditlog	트랜젝션 정보를 audit log(SecAuditLog 지시자에 의해 파일명 지정)에 기록한다.
noauditlog	트랜젝션 정보를 audit log에 기록하지 않는다.
logparts	룰에 따라 로깅 되는 정보를 추가, 삭제 혹은 다른 것으로 바꿀 수 있다. 예) SecFilter 111 pass, logparts:ABCDEFZ SecFilter 222 pass, logparts:+C SecFilter 333 pass, logparts:-F

SecFilterSignatureAction "deny, log, status:403"

룰 설정을 유지하기 쉽게 만들어 주는 이 지시자는 1.9RC1 버전부터 사용할 수 있다. 이 지시자는 단독 설정 내에서 몇 번이고 쓰일 수 있고, 쓰인 위치의 다음에 나오는 룰에 바로 적용이 된다.
각 룰마다의 동작을 설정할 수 있게 해주는 기능을 한다. 기본값은 “deny,log,status:403"으로 SecFilterSelective 등의 옵션을 사용할 시 SecFilterSiganatureAction 값을 설정해주지 않으면 기본값으로 동작하게 된다. 기본적인 설정 방법은 SecFilterDefaultAction 과 동일하다.

SecFilterCheckURLEncoding On

특수문자들은 URL에 전송되기 전에 인코딩될 필요가 있다. %XY(XY는 16진수) 형태의 문자들은 일반 텍스트 문자로 변환된다.

2.2.2 사용자 Rule 정의

필터링 엔진이 활성화되면 유입되는 모든 요청이 웹서버에 의해 처리되어지기 전에 가로채어지고 분석된다. 앞서 환경설정 지시자들에 의해 웹 요청 형태가 유효한지 등이 점검된 후, 두 번째 단계로 웹 요청은 일련의 사용자 정의 필터를 거치게 된다. 사용자에 의해 정의될 수 있는 대표적인 필터들은 다음과 같다.

SecFilter KEYWORD [ACTIONS]

가장 단순한 형태의 필터링으로 특정 키워드에 의한 필터를 정의할 수 있다. SecFilter 지시자는 웹 요청의 첫 번째 라인에서 특정 키워드가 일치하는지 점검하고, "SecFilterScanPOST On" 설정이 되어 있을 경우에는 body까지 점검한다. 이 때 키워드는 대소문자를 구분하지 않는다.

만일 "SecFilter /bin/sh"와 같이 디렉토리를 포함한 키워드 필터링을 설정해 놓았을 경우 공격자는 "/bin/./sh"와 같이 필터를 우회하여 공격할 수 있다.

따라서, ModSecurity에서는 다음과 같이 공격자가 우회할 수 있는 특정 문자열을 자동으로 변환하여 키워드 필터링을 우회할 수 없도록 하고 있다.

변환 전	변환 후	비고
\	/	윈도우 시스템에서 적용
/./	/	
//	/	
URL 인코딩된 문자열	URL 디코딩된 문자열	

키워드는 단순한 text가 아닌 정규 표현식으로 다양한 필터 규칙을 만들어 적용시킬 수 있다. 키워드에서 "!" 문자를 맨 앞에 넣어서 표현식을 반대로 적용할 수도 있다. 가령, "SecFilter !php" 같이 하여 "php" 문자를 포함하지 않는 모든 요청은 거절할 수도 있다.

SecFilterSelective LOCATION KEYWORD [ACTIONS]

앞서 SecFilter 지시자를 이용한 필터링은 적용이 너무 광범위한 단점이 있다. 이러한 단점을 보완하여 실제 유용하게 사용될 수 있는 지시자가 SecFilterSelective이다. 이 지시자는 SecFilter 지시자에서 LOCATION 부분이 추가 되었는데 해당 키워드를 어느 위치에서 찾을 것인지 지정해서 보다 정확한 필터링을 할 수 있도록 한다. LOCATION 변수는 다음과 같이 파이프(|)로 구분된 일련의 위치 확인자로 구성된다.

```
SecFilterSelective "REMOTE_ADDR|REMOTE_HOST" KEYWORD
```

위의 예는 클라이언트의 IP 주소와 호스트 이름에만 키워드가 적용된다. 위치 확인자에 들어갈 수 있는 것은 다음과 같다.

REMOTE_ADDR	SERVER_ADMIN	TIME_MIN
REMOTE_HOST	SERVER_NAME	TIME_SEC
REMOTE_USER	SERVER_ADDR	TIME_WDAY
REMOTE_IDENT	SERVER_PORT	TIME
REQUEST_METHOD	SERVER_PROTOCOL	API_VERSION
SCRIPT_FILENAME	SERVER_SOFTWARE	THE_REQUEST
PATH_INFO	TIME_YEAR	REQUEST_URI
QUERY_STRING	TIME_MON	REQUEST_FILENAME
AUTH_TYPE	TIME_DAY	REQUEST_BASENAME
DOCUMENT_ROOT	TIME_HOUR	IS_SUBREQ

이 외에도 보다 전문적이고 부가적인 위치 확인자들도 존재하는데 이는 ModSecurity의 매뉴얼을 참조하기 바란다.

SecFilterScanOutput On

아파치 2에서 ModSecurity는 출력 필터를 지원한다. 기본 설정 상, 이 기능은 비 활성화되어 있으므로 위와 같이 활성화시켜 주어야 한다. 기존의 입력 필터는 웹 요청이 아파치에 의해 처리되기 이전에 실행되지만 출력 필터는 아파치에 의해 웹 요청이 처리 완료된 이후에 실행된다.

위와 같이 출력 필터를 설정한 후에 “OUTPUT" 파라메터를 사용하여 특정 키워드를 가진 출력에 대해 필터링을 할 수 있다.

```
SecFilterSelective OUTPUT "Fatal error:" deny,status:500
ErrorDocument 500 /php-fatal-error.html
```

이와 같이 공격자가 공격 정보로 이용될 수 있는 정보나 특정 명령어 실행 결과 등 Critical한 결과가 실행되어 공격자에게 결과가 전달되는 것을 차단할 수 있다.

출력 필터는 일반 평문 text와 HTML 출력에 대해서만 유용하며, 이미지와 같은 바이너리 컨텐츠에 대해 정규식을 적용한다면 서버가 느려질 수 있다. 디폴트로 ModSecurity는 컨텐츠 타입을 가지고 있지 않거나 “text/plan" 또는 ”text/html"을 컨텐츠 타입으로 가진 출력에 대해서만 스캔한다. 스캔하고자 하는 컨텐츠 타입을 바꾸고자 할 경우에는 “SecFilterOutputMimeTypes” 지시자를 이용한다.

```
SecFilterOutputMimeTypes "(null) text/html text/plain"
```

위의 설정은 ModSecurity가 평문 text 파일, HTML 파일 그리고 MIME 타입이 정의되지 않은 파일들에 대해 출력 필터를 적용하게 한다.

출력 필터는 유용한 기능이라고 할 수 있지만 완벽하지는 못하다. 공격자가 모니터링하고 있지 않는 컨텐츠 타입으로 바꾸거나 출력을 인코딩하는 방법으로 필터를 우회할 가능성이 존재한다.

2.2.3. 기타 지시자

SecUploadDir /tmp

ModSecurity는 POST 요청과 multipart/form-data 인코딩을 통하거나 PUT 요청을 통한 파일 업로드를 가로채어 점검할 수 있는 기능이 있다.
ModSecurity는 항상 임시 디렉토리에 파일들을 업로드하게 하는데, 이때 SecUploadDir 지시자를 사용하여 임시 디렉토리를 선택할 수 있다.

SecServerSignature "Microsoft-IIS/5.0"

웹서버는 기본적으로 HTTP 응답에 서버의 정보를 실어서 보낸다. 이 정보는 공격자들이 공격하기 위한 기본 정보가 될 수 있다. 따라서, 이 정보를 SecServerSignature 지시자를 이용하여 바꿈으로써 공격자를 혼란스럽게 할 수 있다.

SecFilterDebugLog logs/modsec_debug_log

SecFilterDebugLog 지시자는 디버깅 결과를 어디에 기록할 것인지를 정의한다. 위치를 지정해 주는 파라메터가 슬래쉬(/)로 시작하지 않으면 아파치 홈 디렉토리로부터의 상대경로를 의미한다.

SecFilterDebugLevel 1

SecFilterDebugLevel 지시자를 사용하여 디버깅 수준을 얼마나 상세하게 수행할 것인지를 결정할 수 있다. 과도한 디버깅 수준은 불필요한 로그를 생산할 수 있으므로 문제 발생 시나 서비스 분석이 필요한 경우 등에 상세 디버깅을 하는 것이 바람직하다.

레벨	설명
0	없음
1	중요 이벤트(error_log에 기록됨)
2	간략한 정보 메시지
3	상세한 정보 메시지

SecAuditEngine On
SecAuditLog logs/audit_log

표준 아파치 로깅은 특정 사용자나 공격자를 추적하기 위해서 부족한 면이 있다. ModSecurity는 SecAuditEngine 지시자를 통해 아파치의 기본 로그 파일(access_log, error_log) 보다 좀 더 상세한 공격 관련 정보를 제공해 줄 수 있다.
SecAuditEngine이 사용할 수 있는 파라메터는 다음과 같다.
- o On : 모든 요청에 대해 로그를 남김
- o Off : 어떠한 요청에 대해서도 로그를 남기지 않음
- o RelevantOnly : 필터에 일치하는 요청에 대해서만 로그를 남김

3. ModSecurity 2.x 설치 및 운영

우리는 앞서 ModSecurity 1.x 버전의 설치 및 운영방법을 간략히 살펴보았다. 이번 절에서는 ModSecurity 2.x 버전의 설치 및 운영 방안에 대해 알아본다.

- o 커널 : Linux 2.4.20-8-686-smp
- o 웹서버 : Apache 2.0.59
- o ModSecurity 소스코드 디렉토리 : /usr/local/modsecurity-apache_2.1.4
- o 아파치 소스설치 디렉토리 : /usr/local/apache2
- o 아파치 웹서버 홈 디렉토리 : /usr/local/apache2/htdocs

ModSecurity가 '08. 2. 20 부로 2.5가 릴리즈 되면서 이 가이드가 최초 개정당시(1월) 안정화 버전인 2.1.4 버전에 비해 많은 발전이 이루어졌다. 이에 따라 ModSecurity의 안정화 버전은 2008년 6월 기준으로 2.1.7('08.4.3 Realesed), 2.5.5('08.6.6)이며 1.x 버전은 1.9.5가 마지막 안정화 버전이다. 본 가이드에서는 2.5.x 버전에 대한 내용은 포함되어 있지 않다.

ModSecurity 설치에 앞서 1.x 버전과 2.x 버전의 차이점을 간단히 알아보자.

구 분	ModSecurity 1.x	ModSecurity 2.x
Apache 버전	Apache 1.x, 2.x 지원	Apache 2.x Only
설치법	apxs이용한 DSO방식, 소스컴파일	Makefile
처리 단계	Inbound(Request body) / Outbound(Response body)	phase:1 (Request header) phase:2 (Request body) phase:3 (Response header) phase:4 (Response body) phase:5 (logging)
설정 방법	apache 1.x 의 경우 <IfModule mod_security.c> apache 2.x 의 경우 <IfModule security_module>	<IfModule security2_module>
룰 활성화	SecFilterEngine	SecRuleEngine
기본 동작	SecFilterDefaultAction	SecDefaultAction
지시자	SecFilter, SecFilterSelective	SecRule
Request body 접근	SecFilterScanPost	SecRequestBodyAccess(phase:2)
Response body 접근	SecFilterScanOutput	SecResponseBodyAccess(phase:4)

※ ModSecurity 2.5.x 버전에 대한 사항은 별도 가이드 제작 예정

ModSecurity 2.x 부터는 Apache 1.x를 지원하지 않는다. 그렇기 때문에 설치 전 시스템의 Apache 버전을 확인 후 설치해야 한다. 그리고 각 버전의 룰이 호환되지 않기 때문에 개별적으로 룰이 생성된다.

3.1 ModSecurity 2.x 다운로드 및 설치

ModSecurity 2.x 부터는 Apache 1.x을 지원하지 않을 뿐 아니라 설치시 apxs를 지원하지 않기 때문에 자체 Makefile을 이용한 설치만 가능하다. 설치하기 앞서 자신의 Apache의 버전을 꼭 확인하고 Apache2가 아니라면 최신 Apache를 먼저 다운로드 후에 설치해야 한다.

Apache는 아래 링크에서 다운로드 받을 수 있다.
http://www.apache.org

이 후, ModSecurity 2.1.4 버전을 아래 링크에서 다운로드 받는다.

- ModSecurity 공식 홈페이지
http://www.modsecurity.org/download/modsecurity-apache_2.1.4.tar.gz

또는,

- KrCERT/CC 공개웹방화벽 페이지
http://www.krcert.or.kr/firewall2/index3.jsp 접속 후 modsecurity-apache_2.1.4.tar.gz 다운

ModSecurity 2.x 버전을 설치하면서 아래 사항을 확인한 뒤 다음 과정을 진행하도록 하자.

	세 부 사 항
PCRE	ModSecurity 2.x가 정상 작동을 위해 필수 설치되 있어야 함. => http://www.pcre.org 다운로드
libxml2	ModSecurity 2.x가 정상 작동을 위해 필수 설치되 있어야 함. => http://www.xmlsoft.org 다운로드
unique_id	ModSecurity 2.x가 필터링 기능을 수행하기 위해 필수 Enable 되있어야 함. => ./configure --enable-unique-id --enable-so Apache 컴파일시 enable해야 하며, RPM 설치 버전은 모듈 로드 확인

위 언급한 세가지 사항을 체크한 뒤 ModSecurity의 설치를 진행하는 것이 수월하다. 다운로드 받은 파일을 압축 해제 하면 다음과 같은 파일과 폴더가 생성된다.

```
# tar xzvf modsecurity-apache_2.1.4.tar.gz
# cd modsecurity-apache_2.1.4; ls -al
합계 52
drwxrwx---    5 1000     1000        4096 11월 28 06:05 .
drwxr-xr-x   22 root     root        4096  1월 16 16:08 ..
-rw-rw----    1 1000     1000        7146 11월 28 03:37 CHANGES
-rw-rw----    1 1000     1000       15128  7월  3  2007 LICENSE
-rw-rw----    1 1000     1000         422  7월 27 21:45 README.TXT
drwxrwx---    3 1000     1000        4096 11월 28 06:05 apache2
drwxrwx---    3 1000     1000        4096 11월 28 06:06 doc
-rw-rw----    1 1000     1000        1880  9월  8 01:24
modsecurity.conf-minimal
drwxrwx---    3 1000     1000        4096 11월 28 06:05 rules
[root@www modsecurity-apache_2.1.4]#
```

ModSecurity 2.x는 Makefile을 이용한 설치방법만 존재하기 때문에 Apache가 설치된 종류에 따라 구분하여 설명한다.

▷ RPM으로 설치된 Apache

Apache가 RPM 패키지로 설치되어 있다면 레드햇 계열의 경우 httpd-devel, 데비안 계열에서는 apache2-prefork-dev나 apache2-threaded-dev 패키지가 설치되 있어야 한다.

① 최초 압축을 해제한 폴더에서 apache2 폴더로 이동하면 Makefile 파일이 존재한다. 이 파일을 열어 다음과 같이 수정한다.

35번째 라인의 top_dir 값 수정
top_dir = /apps/apache22 → top_dir = /etc/httpd

```
top_dir = /usr/share/apache22  // 데비안 계열
top_dir = /var/apache2         // 솔라리스 계열
```

45번째 라인의 libxml2 옵션 확인, 기본적으로 아래 경로에 설치되기 때문에 특이한 경우가 아니라면 수정하지 않아도 된다.
INCLUDES = -I /usr/include/libxml2

② 컴파일하여 설치한다.
make && make install

③ 설치가 완료되면 /usr/lib/httpd/modules/mod_security2.so 파일이 생성되었는지 확인한다. 확인 되었다면 httpd.conf 를 열어 아래 문장을 추가해준다.

```
LoadFile /usr/lib/libxml2.so      //mod_security 보다 먼저 Load되어야 한다.
LoadModule security2_module   modules/mod_security2.so
LoadModule unique_id_module   modules/mod_unique_id.so  // RPM설치이므로 추가
```

④ 아파치 웹서버를 재구동한다.
/etc/init.d/httpd restart

▷ DSO 방식으로 설치된 Apache

① DSO 방식으로 설치된 Apache 역시 Makefile에서 아래 사항을 수정해줘야 한다.

35번째 라인의 top_dir 값 수정
top_dir = /apps/apache22 → top_dir = /usr/local/apache2

45번째 라인의 libxml2 옵션 확인, Default로 아래 경로 설치
INCLUDES = -I /usr/include/libxml2

② 컴파일하여 설치한다.
make && make install

③ 설치가 완료되면 /usr/local/apache2/modules/mod_security2.so 파일이 생성되었는지 확인 한

후 httpd.conf에 아래 문장을 추가해준다.

```
LoadFile /usr/lib/libxml2.so    // mod_security 보다 먼저 Load되어야 한다.
LoadModule security2_module   modules/mod_security2.so
```

재차 강조하지만 소스 컴파일된 Apache의 경우는 unique_id 모듈이 Enable되어 있어야 한다.

④ 아파치 웹서버를 재 구동한다.

```
# /usr/local/apache2/bin/apachectl stop
# /usr/local/apache2/bin/apachectl start
```

3.2 ModSecurity 2.x 활성화 및 Rule 설정

ModSecurity 2.x를 설치하였으니 1.x버전과 마찬가지로 웹 방화벽 기능 적용을 위해 httpd.conf 파일 내에 설정 지시자를 정의해 주자. 2.x버전의 대부분의 지시자는 Virtual Host, Location, Location Match, Directory 등의 Apache 지시자 내에서도 사용할 수 있다. 그 외에 메인파일의 구성 중에서는 오직 한번만 사용할 수 있는 것들이 있는데, 지금부터 자세히 알아보겠다.

마찬가지로 관리의 편의상 별도의 룰 파일을 생성하여 httpd.conf에 Include시키자.

Include conf/mod_security2.conf

ModSecurity 2.x의 룰은 1.x버전과 호환이 되지 않기 때문에 별도로 구분해줘야 한다. 여러 부가기능과 옵션 들이 추가 되었으니 이제 그 업데이트 된 기능을 알아보자.

ModSecurity 2.x는 공식 홈페이지에서 Core-Rule을 제작하여 배포하고 있다. 2.x에서는 다양한 기능이 추가 되었디 때문에 이러한 Core-Rule을 통해 미리 룰의 사용법을 숙지해 두어야 한다. 또한 압축해제하여 생성된 rules 폴더 내에 Core-Rule을 기본 포함되어 있으니 별도로 다운 받을 필요가 없다.

이 외에 KrCERT 홈페이지에서도 Sample Rule을 다운로드 받을 수 있다.

http://www.krcert.or.kr/firewall2/index3.jsp 하단에 업체 및 버전별 Sample Rule 다운

이제 실제로 <IfModule> 태그 안에 들어갈 지시자들 중 기본적인 지시자들을 알아보도록 한다. 상당히 다양한 지시자가 있으므로 자세한 것은 아래를 참고해 보기 바란다.

http://www.krcert.or.kr/firewall2/index2.jsp 에 modsecurity2.1.4-manual.pdf 다운

3.2.1 기본 환경설정 및 로깅 관련 지시자

SecRuleEngine On | Off | DetectionOnly

ModSecurity 기능을 활성화(enable) 시킨다.

- o On : ModSecurity 기능 활성화
- o Off : ModSecurity 기능 비활성화
- o DetectionOnly : 활성화는 하지만 차단하지 않고 탐지만 한다.

SecAuditEngine On | Off | RelevantOnly

감사 로깅에 대한 설정을 구성한다.

- o On : 모든 트랜젝션 로깅
- o Off : 모든 트랜젝션 로깅하지 않음
- o RelevantOnly : Error 또는, Warning 의 트랜젝션, 그리고 SecAuditLogRelevantStatus에 정의된 상태코드와 일치하는 트랜젝션만 로깅

SecAuditLog logs/modsec_audit.log

감사 로그 파일의 경로를 정의한다.

예) SecAuditLog /usr/local/apache2/logs/modsec_audit.log

이 파일은 Root로 접근했을 때만 확인할 수 있어야 하며 Root가 아닌 사용자의 접근은 불가해야 한다. 이 로그 파일에는 Serial 방식의 로깅 형태에 사용된다. 만약 Concurrent 방식의 로깅 형태라면 이 파일은 로그의 Index로 쓰여 지게 되며 모든 감사로그에 대한 파일이 생성되면서 그 기록을 포함하게 된다. 만약 Concurrent 로깅방식을 사용하고 감사 로그를 원격 콘솔로 보내려면 modsec-auditlog-collector.pl 스크립트를 아래와 같이 사용하면 된다.

예) SecAuditLog \
"|/path/modsec-auditlog-collector.pl /path/SecAuditLogDataDir /path/SecAuditLog"

SecAuditLogParts

로그 파일에 기록할 항목을 정의한다.

예) SecAuditLogParts ABCFHZ

Parts	세 부 내 용
A	audit log header **(필수)**
B	request header
C	request body (request body가 존재하고 modsecurity가 request body를 검사하도록 설정되어 있는 경우에만)
D	**보류중**, response header의 중개 (현재 지원 안 됨)
E	response body 중간 단계(현재 modsecurity가 response body를 검사하며 감사 로깅 엔진이 이를 저장하게끔 설정되어 있는 경우에만)

Parts	세 부 내 용
F	최종 response header(마지막 컨텐츠 전달 과정에서 아파치에 의해 매번 추가되는 날짜와 서버 헤더를 제외한)
G	실제 response body(현재 지원 안됨)
H	감사 로그 트레일러
I	이 옵션은 C를 대체하는 옵션이다. multipart/form-data 인코딩이 사용되었을 때를 제외한 모든 경우엔 C와 같은 데이터를 기록한다.
J	**보류중,** 이 옵션은 multipart/form-data 인코딩을 사용하는 파일 업로드에 대한 정보를 포함할 때 효과가 있다.
Z	로그의 끝을 의미한다. **(필수)**

SecAuditLogRelevantStatus REGEX

감사로깅의 목적과 관련된 response 상태코드의 값을 설정한다.
o 매개변수에는 정규표현식이 들어간다.
예) SecAuditLogRelevantStatus ^[45]

SecAuditLogType Serial | Concurrent

감사로깅 구조의 타입을 설정한다.
o Serial - 모든 로그는 메인 로그파일에 저장된다. 일시적으로 편리할 순 있지만 하나의 파일에만 기록되기 때문에 느려질 수 있다.
o Concurrent - 로그가 각 트랜잭션 별로 나누어 저장된다. 이 방식은 로그파일을 원격 ModSecurity Console host로 보낼 때 사용하는 방식이다.

SecDefaultAction "log, auditlog, deny, status:403, phase:2, t:lowercase"

룰이 매칭되면 기본적으로 취할 행동을 정의한다. 1.9.x 버전의 SecFilterDefaultAction을 떠올리면 이해가 쉬운데 2.1.x 버전에서는 옵션들이 더욱 다양해졌다. 룰이 특정 액션들에 대한 개별 룰을 적용하거나 다른 SecDefaultAction이 정의되어있지 않다면 최초 지정된 SecDefaultAction의 설정을 따른다. 위의 예는 룰이 매칭 되었을 때 차단하며 로그를 남기고, 403 상태코드 페이지를 보여주며 필터링 단계는 “2”이다. 기본적으로 대문자는 모두 소문자로 바뀌어 필터링 된다.

SecDefaultAction 지시자의 추가적인 Action에 대해 알아보자. 먼저, 다음과 같은 5가지 종류로 분류할 수 있다.

구 분	설 명
Disruptive actions	ModSecurity가 데이터를 중간에서 가로챌 때 일어나는 행위이다. 하나의 체인에 첫 번째 룰에만 나타낼 수 있다. allow, deny, drop 등이 있다.
Non-Disruptive actions	어디에나 나타낼 수 있다. capture, ctl, exec, initcol 등이 있다.
Flow actions	하나의 체인에 첫 번째 룰에만 나타낼 수 있다. chain 이 있다.
Meta-data actions	하나의 체인에 첫 번째 룰에만 나타낼 수 있다. id, rev, severity, msg가 있다.
Data actions	전면적으로 수동적이고 다른 행위에서 사용된 데이터를 운반하는 역할뿐이다.

앞서 SecDefaultAction 지시자에 의한 기본 적용 action의 예에서는 콤마(,)로 구분된 6개의 action을 정의하고 있다. 취할 수 있는 대표적인 행위는 다음 표와 같다.

<table>
<tr><th>행 위</th><th>설 명</th></tr>
<tr><td>allow</td><td>룰에 매칭되면 처리를 멈추고 트랜젝션을 허가한다.
예) SecRule REMOTE_ADDR "^192\.168\.1\.100$" "nolog, phase:1, allow"
※ allow 액션은 오직 현재 처리단계에서만 적용된다.</td></tr>
<tr><td>auditlog</td><td>감사로그에 트랜젝션을 기록한다.
예) SecRule REMOTE_ADDR "^192\.168\.1\.100$" "auditlog, phase:1, allow"</td></tr>
<tr><td>capture</td><td>정규표현식과 함께 사용되었을 때 capture 액션은 정규표현식의 사본을 만들고 트랜잭션의 가변적인 콜렉션에 모은다. 최대 10개가 수집되면 각각 0~9까지 한 자리 숫자의 이름이 매겨 진다.
예) SecRule REQUEST_BODY "^username=(\w{25,})" "phase:2,capture,t:none,chain"
SecRule TX:1 "(?:(?:a(dmin|nonymous)))"</td></tr>
<tr><td>chain</td><td>두개의 룰을 연관지어 허용하고 싶을 때 사용할 수 있다.
예) SecRule REQUEST_METHOD ^POST$ "chain"
SecRule REQUEST_HEADER:Content-Length ^$</td></tr>
<tr><td>ctl</td><td>ctl 액션은 트랜잭션을 최신화하기 위한 설정 옵션이다.
예) SecRule REQUEST_CONTENT_TYPE ^text/xml "nolog,pass,ctl:auditEngine=ON"
다음과 같은 옵션들이 들어갈 수 있다.
<table>
<tr><td>auditEngine</td><td>requestBodyProcessor</td></tr>
<tr><td>auditLogParts</td><td>responseBodyAccess</td></tr>
<tr><td>debugLogLevel</td><td>responseBodyLimit</td></tr>
<tr><td>requestBodyAccess</td><td>ruleEngine</td></tr>
<tr><td>requestBodyLimit</td><td></td></tr>
</table>
</td></tr>
</table>

행 위	설 명
deny	룰이 일치할 경우 요청 처리를 차단한다. status action이 명시되어 있지 않으면 ModSecurity는 “HTTP 403 error code"를 반환한다. 예) SecRule REQUEST_HEADERS:User-Agent \ "nikto“ ”log,**deny**,msg:’Nikto Scanners Identified’"
drop	FIN 패킷을 보내 TCP 연결을 끊어서 연결을 종료시킨다.
exec	필터가 일치하면 파라미터를 지원하는 외부 스크립트/바이너리를 실행시킨다. 예) SecRule REQUEST_URI "^/cgi-bin/script\.pl" \ "log, **exec:/usr/local/apache/bin/test.sh**, phase:1"
id	룰이나 체인에 unique ID를 할당한다. 예) SecRule &REQUEST_HEADER:Host "@eq 0" \ "log, **id:60008**, severity:2, msg:’Request Missing a Host Header’"
log	필터가 일치하였을 때 로그를 남긴다. ※ 이 동작은 Apache의 error log와 ModSecurity의 audit log를 비교한다.
msg	룰이나 체인에 사용자 메시지를 지정한다.
noauditlog	필터가 일치하였을 때 해당 트랜젝션을 audit log에 기록하는 기준으로 하지 않는다. 예) SecRule REQUEST_HEADER:User-Agent "Test" "allow, **noauditlog"** ※ 만약 SecAuditEngine이 On으로 설정되어 있으면 모든 트랜젝션은 로깅된다. 만약 RelevantOnly로 설정되어 있으면, noauditlog 로 제어할 수 있다. noauditlog가 설정되고 audit event가 발생한다면 트랜젝션을 로깅할 수 있다. 확실하게 audit logging을 비활성화 하려면 "ctl:auditEngine=Off"를 쓰면 된다.
nolog	에러와 감사로그에서 나타나는 일치 현상을 방지한다. 예) SecRule REQUEST_HEADERS:User-Agent "Test" "allow, **nolog"**
pass	필터에 일치해도 요청을 그냥 허용하고 다음단계로 진행한다. 예) SecRule REQUEST_HEADERS:User-Agent "Test" "log,**pass**"
pause	요청에 대한 응답을 하기 전에 정의된 수 milliseconds 동안 중지시킨다. 예) SecRule REQUEST_HEADERS:User-Agent "Test" "log,deny,**pause:5000"** ※ Brute Force scanner의 속도를 떨어트리거나 보호할 수 있다. DoS 유형의 공격아래에 있을 때는 pause가 완료할 때까지 child process가 무력화되는 경우가 발생할 수 있다.
phase	룰이나 체인이 속하게 되는 1~5 까지의 처리 단계 예) SecDefaultAction "log,deny,**phase:1**,t:removeNulls,t:lowercase" SecRule REQUEST_HEADERS:User-Agent "Test" "log,deny,status:403" ※ 잘못된 phase를 지정하지 않게 유의해야 한다. 대상 변수가 무의미 할 수 있다. 변수나 정규식이 false negative를 일으킨다면 올바른 작동이다. 그러나 악성 데이터를 처리하지 못하는 것은 잘못된 phase를 지정했기 때문이다.

행 위	설 명
proxy	필터가 일치하면 트랜젝션을 가로채어 프록시가 구동중인 다른 웹서버로 요청을 포워딩한다. 예) SecRule REQUEST_HEADERS:User-Agent "Test" \ "log, **proxy:http://www.honeypothost.com**" ※ 이 동작은 mod_proxy가 반드시 설치되 있어야 한다.
redirect	필터가 일치하면 사용자를 주어진 URL로 리다이렉트 시킨다. 예) SecRule REQUEST_HEADERS:User-Agent "Test" \ **"redirect:http://www.krcert.or.kr/failed.html"** ※ 만약 status 액션이 현재 사용하고 있거나 가능한 값이 301,302,303,307 이라면 그것은 redirection에 사용될 것이다. 그렇지 않다면 302 상태 코드를 사용한다.
skip	필터가 일치하면 하나 이상의 룰이나 체인을 건너뛸 수 있다. 예) SecRule REQUEST_URI "^/$" "chain,**skip:2**" SecRule REMOTE_ADDR "^127\.0\.0\1$" "log,deny,status:400" SecRule &REQUEST_HEADERS:Accept "@eq 0" \ "log,deny,log,status:400,msg:'Request Missing an Accpet Header'"
status	deny 나 redirect 액션에서 사용될 상태코드 값을 지정한다. 예) SecDefaultAction "log,deny,**status:403**,phase:1"
t	이 액션은 룰에 지정된 연산자가 실행되기 전 변수에 지정된 변형 함수에 사용될 수 있다. 예) SecDefaultAction "log,deny,phase:1,t:removeNulls,t:lowercase" SecRule REQUEST_COOKIES:SESSIONID "47414e81....." \ "log, deny,status:403,**t:md5**"

이 외에도 추가된 더 다양한 기능 및 옵션들이 있으나 여기서 다루기에는 그 양이 방대하다 보니 간략한 설명으로 마친다. ModSecurity 1.x에서 지원되는 기능들은 거의 그대로 따르고 있지만 그 쓰임이 조금 변했거나 더 강화된 차이가 있다.

SecRequestBodyAccess On | Off

Request 값에서 Body 부분에 대한 처리를 어떻게 할 것인지 구성한다.
- o On : RequestBody 접근을 시도한다.
- o Off : RequestBody 접근시도를 하지 않는다.

이 지시자는 Request 값에서의 POST_PAYLOAD를 검사할 때 필요하다. POST값을 필터링하기 위해서는 phase:2와 REQUESET_BODY 변수/로케이션, 3가지가 모두 구성되어야만 처리가 가능하다.

SecRequestBodyLimit

ModSecurity가 Request Body 크기로 할당할 수 있는 메모리 최대 크기를 설정한다.

o SecRequestBodyLimit 134217728

131072KB (134217728 byte)가 기본값이다.

SecReponseBodyAccess On | Off

Response 값에서 Body 부분에 대한 처리를 어떻게 할 것인지 구성한다.

o On : ResponseBody 접근을 시도한다. (그러나 MIME 타입과 일치해야만 한다.)

o Off : ResponseBody 접근시도를 하지 않는다.

이 지시자는 html 응답을 조사하기 위해 필요하다. "phase:4"의 처리 단계와 RESPONSE_BODY 변수/로케이션, 3가지가 설정되어 있지 않으면, response body를 검사할 수 없다.

SecResponseBodyLimit 524228

ModSecurity가 Response Body 크기로 할당할 수 있는 메모리 최대 크기를 설정한다.

o SecRequestBodyLimit 524228

이 값을 넘어가면 서버는 500 내부 서버 오류 메시지만 표시할 것이다.

SecReponseBodyMimeType mime/type

Response 값에서 Body 값을 버퍼링할 MIME 타입을 설정한다.

o SecResponseBodyMimeType text/plain text/html // 기본값

Mime 타입은 복수로 추가할 수 있다.

SecReponseBodyMimeTypesClear

ResponseBody의 버퍼링을 위해 Mime 타입의 목록을 지우며, 처음에 위치시킨다.

o SecResponseBodyMimeType

3.2.2 사용자 Rule 정의

SecRule REQUEST_URI "attack"

SecRule은 ModSecurity의 핵심 지시자이다. 데이터 분석과 액션 실행의 기본이 된다.
 o SecRule REQUEST_URI "attack"

SecRule은 다음 세가지로 구성되어 있다.
 o SecRule VARIABLES OPERATOR [ACTIONS]
  - Operator는 어떻게 검사할 것인지를 지정하며,
  - Actions는 operator을 통해 변수가 실행되어 일치하였을 때 어떤 행동을 취할 것인지를 지정

① 변수(Variables)
 다음 예제는 'dirty'를 포함한 트랜젝션을 거부한다.
 ex) SecRule **REQUEST_URI** dirty
 중복하여 사용할 수도 있다.
 ex) SecRule **REQUEST_URI|QUERY_STRING** dirty

② 연산자(Operator)
 연산자에는 정규식을 사용할 수 있다. 첫문자는 @ 로 시작하여야 하며 정규식을 이용해 특별한 패턴으로 대상을 검사할 수 있다.
 ex) SecRule REQUEST_URI "**@rx** dirty"
 부정을 표현할땐 다음과 같이 사용할 수 있다.
 ex) SecRule REQUEST_URI "**!@rx** ^0$"

③ 행위(Actions)
 행위는 default action에서 지정되기 때문에 생략할 수 있다. 만약 생략하지 않고 추가로 지정된다면 default action과 병합되어 action이 작용되게 된다. 액션은 위에서 설명하였다.

SecRuleInheritance On | Off

상위 문맥으로부터 룰을 상속받을지 안 받을지 설정하기 위한 지시자이다.
 o SecRuleInheritance On
각 가상호스트마다 별도로 룰을 설정할 수 있다.

```
<VirtualHost *:80>
...
ServerName app1.com
SecRuleInheritance Off
SecDefaultAction log,deny,phase1,redirect:http://www.test2.com
...
</VirtualHos>
<VirtualHost *:80>
...
ServerName app2.com
SecRuleInheritance On SecRule ARGS "attack"
...
</VirtualHos>
```

SecRuleRemoveById

상위 문맥과 룰이 일치하면 일치한 룰을 제거한다.

o SecRuleRemoveById 1 2 "9000-9010"

다중 파라미터를 지원하며 쌍따옴표를 통한 범위를 지정할 수 있다. 파라미터는 RULE ID나 하나의 범위가 될 수도 있다.

SecRuleRemoveByMsg

상위 문맥과 룰이 일치하면 일치한 룰을 제거한다.

o SecRuleRemoveByMsg "FAIL"

위 지시자는 RULE ID로 룰을 제거하지만 이 지시자는 action의 msg가 지정한 정규식에 따라 룰을 제거한다.

SecServerSignature "Netscape-Enterprise/6.0"

응답 헤더의 "Server:" 토큰을 바꿔 표시한다.

o SecServerSignature "Microsoft-IIS/5.0"

이 지시자를 사용하려면 Apache의 설정파일에서 ServerTokens 값이 “Full"로 설정되있어야 한다.

SecDebugLog /path/to/modsec-debug.log

ModSecurity의 디버그 로그의 경로를 설정한다.

o SecDebugLog /usr/local/apache2/logs/modsec-debug.log

SecDebugLogLevel |0|1|2|3|4|5|6|7|8|9

디버그 로그의 상세 수준을 결정한다.

o SecDebugLogLevel 4

level 1~3은 항상 Apache의 에러로그로 보낸다. level 5가 디버깅에 가장 유용하며 그 이상은 서버의 성능을 저하시키기 때문에 권장하지 않는다.

0	로깅하지 않는다.	4	트랜젝션의 처리과정 상세
1	에러만(요청값) 로깅	5	4와 동일하나 각 처리의 부분별 정보가 포함
2	경고 수준	9	모두 로깅, 디버깅 정보를 상세히 포함
3	공지 수준		

SecDataDir /path/to/dir

영구적 데이터를 저장할 경로를 지정한다.(IP주소, Session 값 등)

o SecDataDir /usr/local/apache2/logs/data

이 지시자는 initcol, setsid, setuid 가 사용될 때 필요하다. 반드시 웹서버 사용자에 의해 쓰기가 가능해야 한다.

SecTmpDir /path/to/dir

임시 파일이 생성될 디렉토리를 설정한다.
o SecTmpDir /tmp
Apache 사용자 프로세스에 의해 쓰기가 가능해야 한다. 이 디렉토리는 검사가 진행되는 동안 아파치가 메모리가 부족할 때 디스크를 스왑하기 위한 영역이다.
(메모리 크기는 SecRequestBodyInMemoryLimit 지시자에 의해 설정된 크기이다.)

SecUploadDir /path/to/dir

가로챈 파일을 저장하기 위한 디렉토리를 설정한다.
o SecUploadDir /tmp
SecTmpDir 지시자에 의해 지정된 디렉토리와 같은 파일시스템에 위치해야 한다. 이 지시자는 SecUploadKeepFiles 와 함께 쓰인다.

SecUploadKeepFiles On | Off | RelevantOnly

트랜젝션이 처리된 후에 가로챈 파일을 유지할 것인지 아닌지를 설정한다.
o On : 업로드된 파일을 유지한다.
o Off : 업로드된 파일을 유지하지 않는다.
o RelevantOnly : 수락된 요청과 관련된 업로드 파일만 유지한다.

3.3 트래픽 감사 예외 IP 설정

외부에서 취약점 점검이나 모의 해킹 등을 수행 할 때에는 점검 트래픽과 공격트래픽이 거의 유사하기 때문에 공격으로 탐지되어 차단된다. 그러므로 이러한 진단작업을 진행하기에 앞서, 감사 예외 IP로 설정하면 지정된 IP에 대해서는 차단조치하지 않기 때문에 전상적인 점검이 가능하다.

다음은 본 설정방법을 설명하기 위한 예시 IP 목록이며 아래의 IP들에 대해 감사 예외 IP로 설정하는 방법을 알아본다.

① 221.149.161.5	② 211.252.151.24	③ 211.252.151.44

이 IP 목록을 ModSecurity 설정 파일에서 버전별로 아래와 같이 입력해주면 된다.

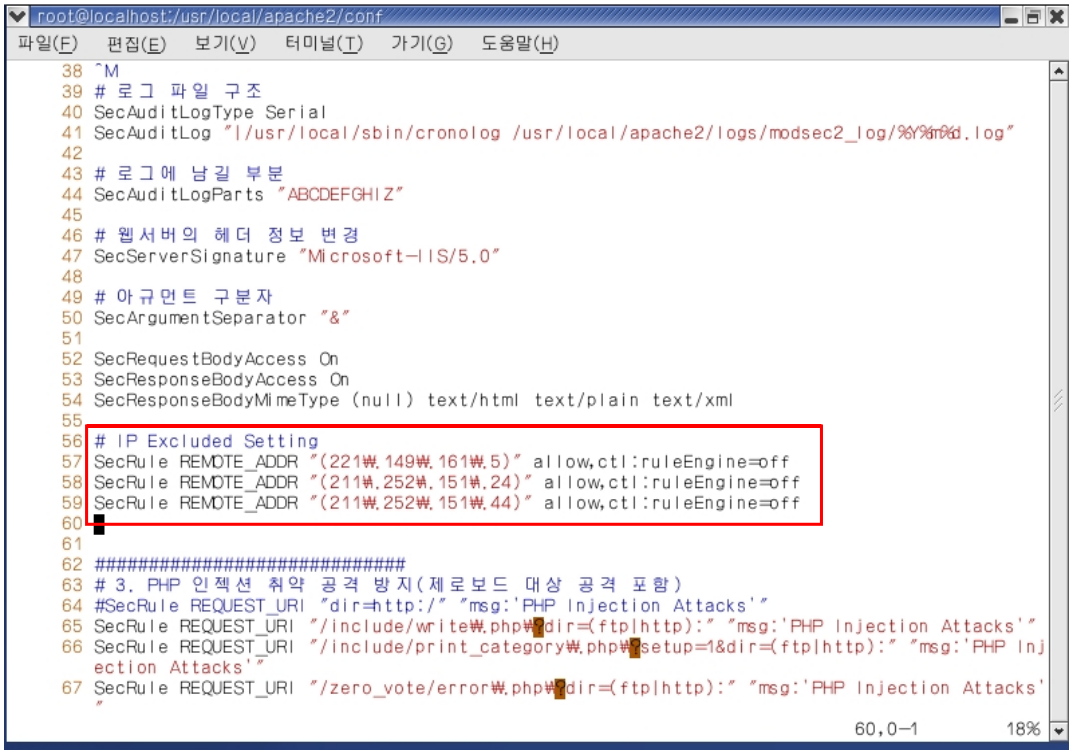
o ModSecurity 1.9.x 버전

형 식	SecFilerSelective REMOTE_ADDR "허용할 IP" allow
사 용	SecFilerSelective REMOTE_ADDR "221\.149\.161\.5" allow SecFilerSelective REMOTE_ADDR "211\.252\.151\.24" allow SecFilerSelective REMOTE_ADDR "211\.252\.151\.44" allow

o ModSecurity 2.x 버전

형 식	SecRule REMOTE_ADDR "허용할 IP" allow,ctl:ruleEngine=off
사 용	SecRule REMOTE_ADDR "221\.149\.161\.5" allow,ctl:ruleEngine=off SecRule REMOTE_ADDR "211\.252\.151\.24" allow,ctl:ruleEngine=off SecRule REMOTE_ADDR "211\.252\.151\.44" allow,ctl:ruleEngine=off

위 설정은 예외처리를 해주지만 그에 대한 로그는 동일하게 기록된다. 취약점 점검 서비스를 통해 다량의 로그 발생시에는 Action 지시자에 nolog를 추가해주면 로그가 기록되지 않는다.

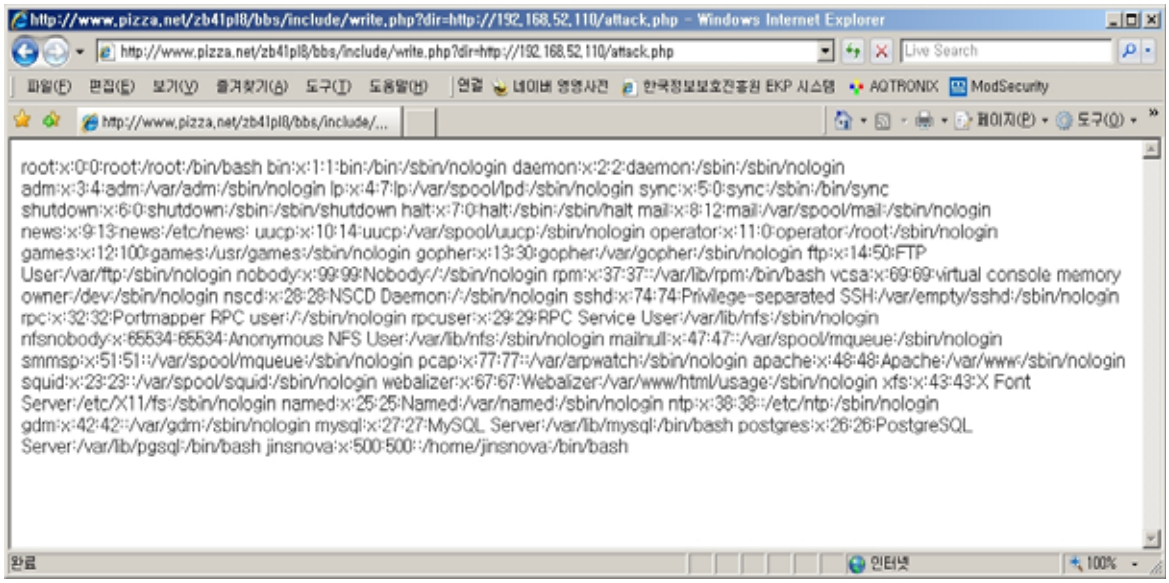


4. ModSecurity를 이용한 모의 공격 차단

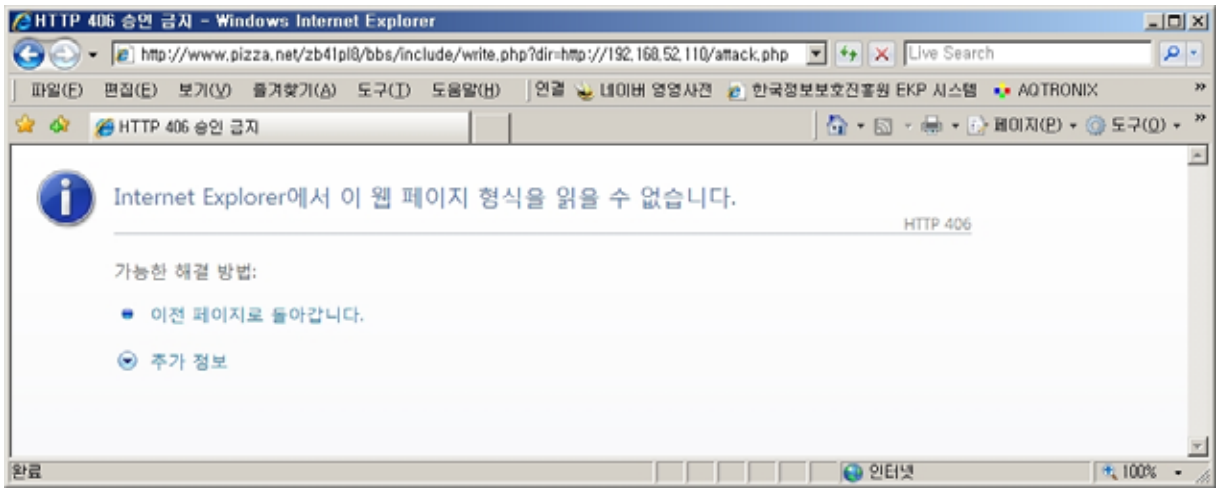
ModSecurity 설치를 완료한 뒤 룰 설정까지 끝났다면 정상적으로 동작하는지 확인해 보자.

다음 화면은 가상 서버에 운영 중인 배달 서비스 홈페이지 화면이다.

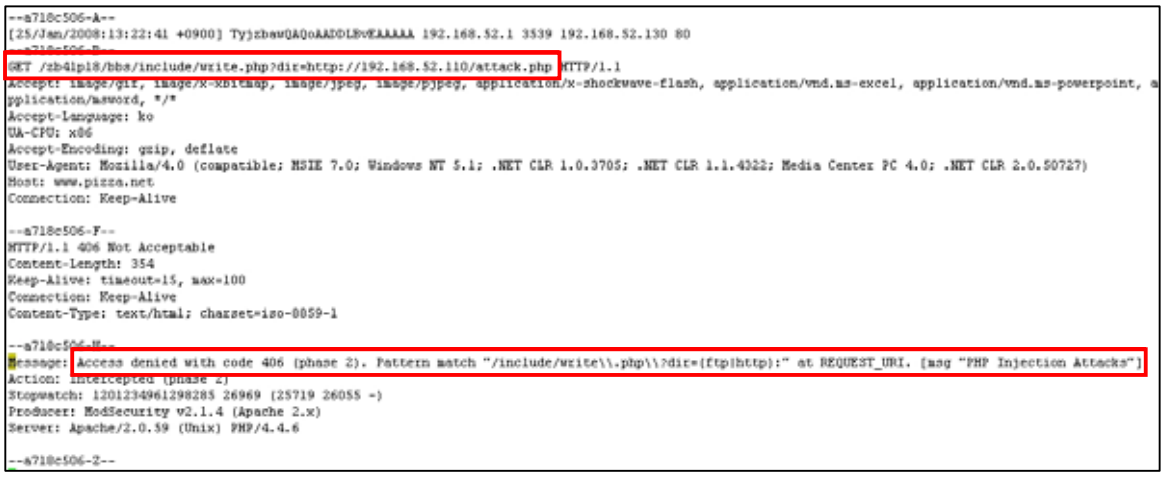
PHP환경에서 제로보드를 설치하여 운영 중인 이 사이트는 제로보드 PHP Injection 취약점에 노출되어 있는 상태였다. 아래 화면은 임의의 서버에서 공격 스트링을 포함한 파일을 Injection 시도하여 대상 서버의 passwd 파일을 출력시킨 화면이다.



이 후 ModSecurity 를 설치하여 동일한 공격 테스트를 시도했더니 아래와 같은 406 에러를 화면에 띄우면서 정상적으로 차단이 되었다.



그리고 아래는 차단된 후 ModSecurity에 의해 생성된 로그 내용이다. 네모박스를 살펴보면 입력된 값과 매치된 rule, 차단 메시지를 볼 수 있다.



이와 같이 ModSecurity는 PHP Injection 이외에도 SQL Injection, XSS, Directory Traversal 공격 등 다양한 웹 해킹 공격으로부터 서버를 안전하게 보호할 수 있다.

공개 웹방화벽인 ModSecurity의 특성상 이번 가이드에서 안내한 기능 외에도 다양한 커뮤니티 또는 포럼 등에서 알려진 기능 및 추가 호환되는 환경에서의 사용법 등이 많이 있기 때문에 KISA에서도 이러한 기능 들을 제공할 수 있도록 많은 노력을 기울일 예정이다. 또한, ModSecurity의 공식 홈페이지에서도 지속적인 Core Rule 업데이트와 버전업이 이뤄지고 있다.

웹 보안의 기본은 Secure Coding과 서버 관리자의 보안 마인드에서부터 비롯된다. 아무리 웹방화벽이나 솔루션들의 기능이 좋다 하더라도 궁극적으로 위 두 가지 사항을 간과한다면 웹사이트의 보안수준 향상에 더욱 더 멀어지게 될 것이다.
이 외에도 KISA에서 발간된 홈페이지 보안가이드나 템플릿을 참고하면서 안전한 웹서버 운영에 도움이 되길 기대한다.